情報処理学会論文誌 Vol.57 No.6 1524–1538 (June 2016)

# 組込システム用マイクロコントローラのための内蔵周辺モジュール割当ての高速化

角川 裕次[1,a)]

受付日 2015年9月3日, 採録日 2016年2月8日

**概要**：近年の組込システム用マイクロコントローラには多種類の周辺モジュールを複数個内蔵しているものが多い．内蔵の周辺モジュールには 1 つあるいは複数の内部のピン（周辺ピン）が付随し，マイクロコントローラデバイスの外部のピン（デバイスピン）と結合されることで，周辺モジュールは外界と信号のやりとりを行う．いくつかのマイクロコントローラには，この結合関係を再マップ可能とする機能を有する．組込システム設計の際には，要求する内蔵周辺モジュールの集合に対して，周辺モジュールのインスタンスを選択し，それらに付随する周辺ピンとデバイスピンの結合を決定しなければならない．この問題は組合せ探索問題であり，要求する周辺モジュールの数が増えると探索空間は爆発的に大きくなる．本論文では，この課題を新たな問題として提案・定式化を行い，割当てに関して互換性のあるデバイスピンをグループ化する手法を提案し，組合せ数を減少させて探索を高速化する．提案手法はバックトラック法による探索，および ZDD 構築アルゴリズムに対して適用して，その有効性を実験的に示した．

**キーワード**：組合せ解の列挙，組込システム，マイクロコントーラ，周辺モジュール，バックトラック，ZDD

# Fast Assignment of Built-in Peripheral Modules for Embedded System Microcontrollers

Kakugawa Hirotsugu[1,a)]

Received: September 3, 2015, Accepted: February 8, 2016

***Abstract:*** Recent microcontrollers for embedded systems have several number of instances of built-in peripheral modules of many kind. Built-in peripheral modules have one or more internal pins (“peripheral pins”), and they are tied to external pins (“device pins”) for interfacing with the external world. Some microcontrollers have remapping features for ties between peripheral pins and device pins. When an embedded system is designed, we must select instances of built-in peripheral modules, and must decide ties between peripheral pins and device pins, under constraints of possible ties. This is a combinatorial problem, and computational cost is extremely high. In this paper, we formulate this new problem, and we propose a method to accelerate finding solutions by grouping compatible device pins. The proposed method is applied for a backtrack-based algorithm and a ZDD construction algorithm, and it is evaluated by computer experiments.

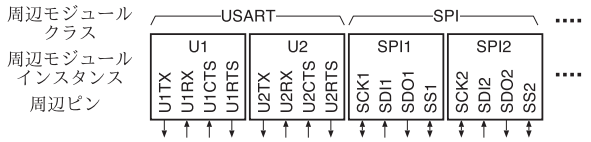
***Keywords:*** enumeration of combinations, embedded system, microcontroller, peripheral module, backtracking, ZDD

## 1. はじめに

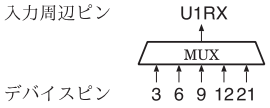
### 1.1 背景

近年の組込システム用マイクロコントローラには多種の周辺モジュールを内蔵しているものが多く，外部インタフェースのための外付け回路を減少できる．マイクロコントローラ内蔵の周辺モジュール（以下では**周辺モジュール**と呼ぶ）には，1 つあるいは複数のデバイス内部のピン（以下では**周辺ピン**と呼ぶ）が付随している．また，マイクロコントローラのパッケージには，物理的な外部のピン（以下では**デバイスピン**と呼ぶ）が装備されている．周辺ピンとデバイスピンが結合されることで，周辺モジュールは外界と信号のやりとりを行う．本論文では周辺モジュール

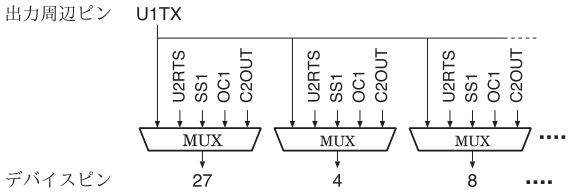
[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] kakugawa@ist.osaka-u.ac.jp



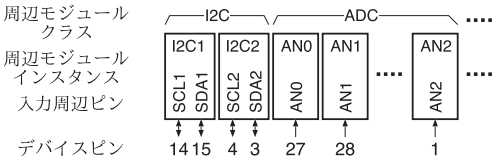

(a) 周辺モジュールと周辺ピンの例




(b) 再マップ機能を有する入力周辺ピンの例




(c) 再マップ機能を有する出力周辺ピンの例




(d) 再マップ機能を有しない周辺ピンの例

**図 1** PIC32MX170F256B における
周辺モジュール，周辺ピン，デバイスピンの例

**Fig. 1** Examples of peripheral modules, peripheral pins, and device pins of PIC32MX170F256B.

に関して以下の特徴を持つマイクロコントローラを対象とする（**図 1** 参照）[*1]．なお特に断らない限り，本論文で示す具体例は Microchip Technology 社 PIC32MX170F256B [6] での場合を示す．

- 同種の周辺モジュールのインスタンスを複数個有する．たとえば，非同期シリアル通信（USART）モジュールのインスタンス U1 と U2 の 2 つを内蔵し，それらを互いに独立して同時に利用できる（図 1 (a) 参照）．
- 一部の周辺ピンは，結合するデバイスピンを変更できる**再マップ**（**remap**）機能を有する．たとえば，非同期シリアル通信モジュール U1 のデータ受信用の周辺ピン U1RX は，マルチプレクサ（MUX）の設定により，デバイスピン 3，6，9，12，21 のどれか 1 つと選択的に結合できる（図 1 (b) 参照）．

なお，以下の 2 点にも注意されたい．(1) デバイスピンの中には，結合可能な周辺ピンが複数あるものが存在する（図 1 でのデバイスピン 3 や 4 等）．この場合でも，1 つのデバイスピンに複数の周辺ピンを同時に結合できない．(2) 再マップ機能を持たずに，結合されるデバイスピンが 1 つに固定されている周辺ピンが存在する（図 1 (d) 参照）．

[*1] この特徴を有するものの例として，Microchip Technology 社 PIC16(L)F，PIC18F，PIC24F，PIC32MX，PIC32MZ 各シリーズの他，STMicroelectronics 社 STM32F0 シリーズのマイクロコントローラが同様な機構を有している．

組込システム設計の際には，たとえば，非同期シリアル通信モジュール 1 つ，AD 変換 4 チャンネル，PWM 出力 2 チャンネル，という要求に対し，各デバイスピンが重複して割り当てられないよう，各周辺モジュールのインスタンスを選び，周辺ピンとデバイスピンの結合（再マップ）関係を決定しなければならない．この問題は，制約を満たす組合せ解の探索を行う問題である．しかし再マップ機能での選択肢が多いマイクロコントローラでは探索空間は広大であり，限られた時間とメモリで制約を満たす解を見つけることは困難である[*2]．本研究の最終的な目標は，複数ある割当て解の中から，配線コストを最小化する解の発見である．本論文ではこれに向けた第 1 歩として，解の列挙の高速化に取り組む．本論文の貢献は，この問題を新たな問題として提案・定式化し，高速化手法の考案・実装評価を行う点にある．

### 1.2 関連研究

過去に同じ課題に取り組んだ研究は著者の知る限り存在しないため，以下では，組合せ探索問題における一般的な関連研究について述べる．

状態空間内で条件を満たす組合せ解の探索は計算機の重要な応用の 1 つであり，古来より多くの研究がなされてきている．中でもバックトラック法は古典的かつ基本的な探索法である．バックトラック法により探索・列挙を行えるものの，組合せ爆発により多大な時間を要し，現実的な時間内では不可能な場合も多い．

グラフの 2 頂点間を結ぶ疎な経路の列挙を求める問題 [11] に対し，解の列挙を表現する ZDD（Zero-suppressed Binary Decision Digram; ゼロサプレス型 2 部決定グラフ）を高速に構築するアルゴリズムを，文献 [5] で Knuth が，文献 [13] で川原らが，それぞれ提案している．いずれのアルゴリズムも，次に探索する状態と等価でしかも探索済みの状態があれば，探索済みの状態を共有することで，探索を高速化している．

一方，列挙問題ではなく，条件を満たす組合せ解を 1 つ発見する問題も重要であり，その例としてモデル検証があげられる [2], [4]．厳密な検証においては，次に訪問しようとする状態が検査済みであるか否かを記憶しておく必要がある．しかし対象が大規模な場合ではメモリや時間の制約から厳密な検証はできず，検査する状態の被覆率を 100%未満にせざるを得ない場合がある．これを行う手法の例として，検査済み状態の記憶に偽陽（false-positive）性を許して使用メモリ量を削減する，ビット状態ハッシュ法（bitstate hashing）あるいは超追跡法（supertrace）や，ブルームフィルタ（Bloom filter）があげられる [4], [10]．

[*2] デバイスを決めれば解が存在する要求は有限個であるが，その数は非常に大きく事前計算は事実上不可能である．

### 1.3 本論文の意義と内容

本論文の成果の意義は以下のとおりである．

- 勘に基づく手作業での解の発見は著しく困難であり，計算機を用いた自動的な探索が重要である．しかし現実的な時間で解が発見できなければ，組込システム開発の支援となり得ない．
- 解が存在しないときはそのことを高速に知ることも重要である．なぜなら，要求を満たす最小規模のマイクロコントローラの判別が容易となり，基盤面積と部品コストの削減に寄与できるからである．

本論文での提案手法の基本アイディアは，割当て対象を同値類分割して組合せ数を減少させるものである．本論文における主要な貢献は，対象としている問題での同値類分割の手法の提案である．提案手法の適用により，同等で別の形の組合せ解の列挙問題のインスタンスを得るが，これに対してさらに既存の高速化手法を適用可能である．また提案手法を，偽陽性を許して訪問済み状態を記憶する手法にも適用可能である．これらのことから，既存の高速化手法については，本論文での議論の対象としない．

### 1.4 本論文の構成

本論文の構成は以下のとおりである．2 章では，準備として，本論文で想定するマイクロコントローラの周辺モジュールに関する事項を抽象化し，問題の定義を行う．3 章では，素朴な探索アルゴリズムを示す．具体的には，バックトラック法による解の列挙アルゴリズムと，解の列挙を表現する ZDD を構築するアルゴリズムを示す．4 章では，互換性のあるデバイスピンをグループ化して高速化する，互換デバイスピングループ（CDPG）の概念を提案し，2 種のアルゴリズムへの適用を行う．5 章では，提案手法の評価結果を示す．6 章では，本研究で得られた結果のまとめを行い，今後の研究課題について述べる．

## 2. 準備

本章では，本論文で取り組む問題の形式的な定義を行う．その準備として，想定するマイクロコントローラの周辺モジュールとそれに関連するものを抽象化する各種の記号を定義する．なお，1 つの記号を（類似した）複数の意味で用いる場合があるので注意されたい．たとえば定義 9 と定義 13 では制約の付け方が異なるが，ともに記号 $B$ を用いて周辺ピンの（部分）集合を表している．本論文で使用する主な記号を**表 1** に掲げる．なお以下では，任意の集合 $U$ に対する $U$ の冪集合，すなわち $\{U' : U' \subseteq U\}$ を，記法 $\mathcal{P}(U)$ で表す．

**表 1** 本論文で使用する主な記号
**Table 1** Symbol table.

| 記号 | | 意味 | |
|---|---|---|---|
| $\mathcal{A}$ | | デバイスピンすべての集合 | 定義 1 |
| $a, a_i, \ldots$ | $\in \mathcal{A}$ | デバイスピンを表す変数 | 定義 1 |
| $\mathcal{B}$ | | 周辺ピンすべての集合 | 定義 2 |
| $b, b_i, \ldots$ | $\in \mathcal{B}$ | 周辺ピンを表す変数 | 定義 2 |
| $A(b)$ | $\subseteq \mathcal{A}$ | デバイスピンの集合で，周辺ピン $b$ に結合可能なもの | 定義 3 |
| $\mathcal{C}$ | | 周辺モジュールクラスすべての集合 | 定義 4 |
| $c, c_i, \ldots$ | $\in \mathcal{C}$ | 周辺モジュールクラスを表す変数 | 定義 4 |
| $\mathcal{M}$ | | 周辺モジュールインスタンスすべての集合 | 定義 5 |
| $m, m_i, \ldots$ | $\in \mathcal{M}$ | 周辺モジュールインスタンスを表す変数 | 定義 5 |
| $C(m)$ | $\in \mathcal{C}$ | 周辺モジュールクラスで，周辺モジュールインスタンス $m$ が属するもの | 定義 6 |
| $\mathcal{F}$ | | 周辺モジュール機能すべての集合 | 定義 7 |
| $f, f_i, \ldots$ | $\in \mathcal{F}$ | 周辺モジュール機能を表す変数 | 定義 7 |
| $S(f)$ | $\subseteq \mathcal{P}(\mathcal{M} \times \mathcal{P}(B))$ | 周辺モジュール機能 $f$ の仕様 | 定義 8 |
| $B(a)$ | $\subseteq \mathcal{B}$ | 周辺ピンの集合で，デバイスピン $a$ に結合可能なもの | 定義 9 |
| $\mathcal{B}^R$ | $\subseteq \mathcal{B}$ | 周辺ピンの集合で，再マップ可能なもの | 定義 10 |
| $\mathcal{B}^N$ | $\subseteq \mathcal{B}$ | 周辺ピンの集合で，再マップ不可能なもの | 定義 10 |
| $M(c)$ | $\subseteq \mathcal{M}$ | 周辺モジュールインスタンスの集合で，周辺モジュールクラス $c$ に属するもの | 定義 11 |
| $M(f)$ | $\subseteq \mathcal{M}$ | 周辺モジュールインスタンスの集合で，周辺モジュール機能 $f$ で限定したもの | 定義 12 |
| $B(f, m)$ | $\subseteq \mathcal{B}$ | 周辺ピンの集合で，周辺モジュールインスタンス $m$ と周辺モジュール機能 $f$ で限定したもの | 定義 13 |
| $C(f)$ | $\in \mathcal{C}$ | 周辺モジュールクラスで，周辺モジュール機能 $f$ が属するもの | 定義 14 |
| $\mathcal{G}$ | | CDPG 分割でのグループ番号の集合 | 定義 18 |
| $g, g_i, \ldots$ | $\in \mathcal{G}$ | CDPG 分割でのグループ番号を表す変数 | 定義 18 |
| $\mathcal{A}^R$ | $\subseteq \mathcal{A}$ | デバイスピンの集合で，再マップ可能な周辺ピンと結合可能なもの | 定義 19 |
| $\mathcal{A}^N$ | $\subseteq \mathcal{A}$ | デバイスピンの集合で，$\mathcal{A}^R$ に含まれないもの | 定義 19 |
| $\mathcal{B}^{NC}$ | $\subseteq \mathcal{B}^N$ | 周辺ピンの集合で，CDPG デバイスピン結合の再マップ不可能なもの | 定義 21 |
| $\mathcal{B}^{NN}$ | $\subseteq \mathcal{B}^N$ | 周辺ピンの集合で，非 CDPG デバイスピン結合の再マップ不可能なもの | 定義 21 |

### 2.1 基本的な記号の定義

対象とするマイクロコントローラを抽象化する，基本的な記号の定義を行う．

**定義 1** **デバイスピンの集合**を記号 $\mathcal{A}$ で表記し，その各要素を記号 $a_0, a_1, ....$ あるいは $a$ で表記する． □

**例 1** $\mathcal{A} = \{1, 2, 3, ..., 28\}$ □

**定義 2** **周辺ピンの集合**を記号 $\mathcal{B}$ で表記し，その各要素を記号 $b_0, b_1, ...$ あるいは $b$ を用いて表記する． □

**例 2** $\mathcal{B}$ = {SDA1, SCL1, SDA2, SCL2, U1TX, U1RX, U1CTS, U1RTS, U2TX, U2RX, U2CTS, U2RTS, AN1, AN2, AN3} □

**定義 3** **周辺ピンとデバイスピンの結合可能関係**を表す写像 $\mathcal{B} \to \mathcal{P}(\mathcal{A})$ を記号 $A$ で表記する．ただし制約 $\forall b \in \mathcal{B} : |A(b)| \geq 1$ を満たす，すなわち，各周辺ピンには少なくとも 1 つの結合可能なデバイスピンが存在するものとする．周辺ピン $b \in \mathcal{B}$ とデバイスピン $a \in \mathcal{A}$ が $a \in A(b)$ の関係にある場合，$b$ と $a$ は**結合可能**といい，それ以外の場合は**結合不可能**という．もし周辺ピン $b \in \mathcal{B}$ に対し $|A(b)| > 1$ の場合，すなわち $b$ が 2 つ以上のデバイスピンの中から選択的に結合可能である場合，$b$ は**再マップ可能**という．$|A(b)| = 1$ の場合，$b$ は**再マップ不可能**という． □

**例 3** $A$(U1TX) = {2, 7, 11, 16, 26}, $A$(SDA1) = {18}（周辺モジュールピン U1TX は再マップ可能であり，SDA1 は再マップ不可能である）． □

**定義 4** **周辺モジュールクラスの集合**を記号 $\mathcal{C}$ で表記し，その各要素は記号 $c_0, c_1, ...$ あるいは $c$ で表記する． □

**例 4** $\mathcal{C}$ = {I2C, U, ADC} □

**定義 5** **周辺モジュールインスタンスの集合**を記号 $\mathcal{M}$ で表記し，その各要素を記号 $m_0, m_1, ...$ あるいは $m$ で表記する． □

**例 5** $\mathcal{M}$ = {I2C1, I2C2, U1, U2, AN1, AN2, AN3} □

**定義 6** **周辺モジュールインスタンスの所属クラス**を表す写像 $\mathcal{M} \to \mathcal{C}$ を記号 $C$ で表記する． □

**例 6** $C$(I2C1) = $C$(I2C2) = I2C, $C$(U1) = $C$(U2) = U, $C$(AN1) = $C$(AN2) = $C$(AN3) = ADC □

**定義 7** **周辺モジュール機能の集合**を記号 $\mathcal{F}$ で表記し，その各要素を記号 $f_0, f_1, ...$ あるいは $f$ で表記する． □

**例 7** $\mathcal{F}$ = {i2c, usart, usart-cts-rts, an} □

$\mathcal{F}$ は周辺モジュールインスタンスとそれに付随する周辺ピン集合で提供する機能の名称の集合である．周辺モジュール機能の概念は次の定義 8 で示すとおり，同一の周辺モジュールインスタンスで異なる使用法ができるよう，使用方法を区別した割当て要求を可能とするものである．

**定義 8** **周辺モジュール機能の仕様**を表す写像 $\mathcal{F} \to \mathcal{P}(\mathcal{M} \times \mathcal{P}(\mathcal{B}))$ を記号 $S$ で表記する．これは与えられた周辺モジュール機能 ($\in \mathcal{F}$) を実現する周辺モジュールインスタンスと周辺ピンの集合の組合せ ($\in \mathcal{M} \times \mathcal{P}(\mathcal{B})$) すべての集合であり，以下の 2 つの制約を満たすものとする．

- 制約 1：$\forall f \in \mathcal{F}\ \forall (m, D) \in S(f)\ \forall (m', D') \in S(f) : C(m) = C(m')$ — 同一の周辺モジュール機能を実現するどの 2 つの周辺モジュールインスタンスも，同一の周辺モジュールクラスに属する．
- 制約 2：$\forall f \in \mathcal{F}\ \forall (m, D) \in S(f)\ \forall (m', D') \in S(f) : (m \neq m') \Rightarrow (D \cap D' = \emptyset)$ — 同一の周辺モジュール機能を実現する異なる周辺モジュールインスタンスは，使用する周辺ピン集合が互いに素である． □

**例 8** $S$(usart) = {(U1, {U1TX, U1RX}), (U2, {U2TX, U2RX})}. $S$(usart-cts-rts) = {(U1, {U1TX, U1RX, U1CTS, U1RTS}), (U2, {U2TX, U2RX, U2CTS, U2RTS})} なお，この例のように異なる周辺モジュール機能が同一の周辺モジュールインスタンスを用いる場合がある． □

なお集合 $\mathcal{A}$, $\mathcal{B}$, $\mathcal{M}$ それぞれにおいて要素間に全順序が定められていると仮定する．また組に対しては，辞書式順序により全順序を定める．

### 2.2 派生的な記号の定義

先述の定義から派生する記号を定め，以降での記述を簡潔にする．

**定義 9** **デバイスピンで制約された周辺ピンの集合**を $B(a) \stackrel{\text{def}}{=} \{b \in \mathcal{B} : a \in A(b)\}$ と定める． □

**例 9** $B(2)$ = {SDA1, AN2, U1RX}, $B(4)$ = {AN3, U1RX} □

**定義 10** **再マップ可能な周辺ピンの集合**を $\mathcal{B}^R \stackrel{\text{def}}{=} \{b \in \mathcal{B} : |A(b)| > 1\}$, **再マップ不可能な周辺ピンの集合**を $\mathcal{B}^N \stackrel{\text{def}}{=} \{b \in \mathcal{B} : |A(b)| = 1\}$ とそれぞれ定める（なお $\mathcal{B}^R$ と $\mathcal{B}^N$ は $\mathcal{B}$ の直和分割である）． □

**例 10** $\mathcal{B}^R$ = {U1TX, U1RX, U2TX, U2RX}, $\mathcal{B}^N$ = {SDA1, SCL1, SDA2, SCL2, AN1, AN2, AN3} □

**定義 11** **周辺モジュールクラスで制約された周辺モジュールインスタンスの集合**を $M(c) \stackrel{\text{def}}{=} \{m \in \mathcal{M} : C(m) = c\}$ と定める． □

**例 11** $M$(I2C) = {I2C1, I2C2}, $M$(U) = {U1, U2} □

**定義 12** **周辺モジュール機能で制約された周辺モジュールインスタンスの集合**を $M(f) \stackrel{\text{def}}{=} \{m \in \mathcal{M} : (m, D) \in S(f)\}$ と定める． □

**例 12** $M$(i2c) = {I2C1, I2C2}, $M$(usart) = {U1, U2}, $M$(usart-cts-rts) = {U1, U2} □

**定義 13** **周辺モジュールインスタンスと周辺モジュール機能で制約された周辺ピンの集合**を $B(f, m) \stackrel{\text{def}}{=} \bigcup_{(m', D) \in S(f) \wedge (m' = m)} D$ と定める． □

**例 13** $B$(usart, U1) = {U1TX, U1RX}, $B$(usart-cts-rts, U1) = {U1TX, U1RX, U1CTS, U1RTS} □

**定義 14** **周辺モジュール機能が属する周辺モジュールクラス**を $C(f) \stackrel{\text{def}}{=} C(m)$ （ただし $(m, D) \in S(f)$）と定

める．□

**例 14** $C(\texttt{i2c}) = \mathsf{I2C}$, $C(\texttt{usart}) = C(\texttt{usart-cts-rts}) = \mathsf{U}$ □

**定義 15** **周辺ピンとデバイスピンの結合の集合**を，$f \in \mathcal{F}, m \in \mathcal{M}$ に対し $D_I(f,m) \stackrel{\text{def}}{=} \{\{(a_{i,1}, b_{i,1}), (a_{i,2}, b_{i,2}), ...\} : \{b_{i,1}, b_{i,2}, ...\} = B(f,m), b_{i,j} \in B(a_{i,j})\}$ と定める（周辺モジュール機能 $f$ を周辺モジュールインスタンス $m$ で実現する場合の，周辺ピンとデバイスピンの結合の組すべての集合を表す）．□

**例 15** $D_I(\texttt{usart}, \mathsf{U1}) = \{\{(2, \mathsf{U1TX}), (9, \mathsf{U1RX})\}, \{(7, \mathsf{U1TX}), (9, \mathsf{U1RX})\}, ..., \{(16, \mathsf{U1TX}), (6, \mathsf{U1RX})\}\}$ □

### 2.3 問題の定義

周辺モジュール割当て列挙問題の定義を示す．

**定義 16** **マイクロコントーラ仕様**は 7 項組（$\mathcal{A}$, $\mathcal{B}$, $A$, $\mathcal{C}$, $\mathcal{M}$, $\mathcal{F}$, $S$）で定める（各要素は先述の記号の定義のとおりである）．□

**定義 17** **マイクロコントローラ周辺モジュール割当て列挙問題**は，与えられたマイクロコントローラ仕様と割当て要求をする周辺モジュール機能の列に対し，制約を満たす割当てを解を列挙する問題である．形式的には，入力，出力，制約は以下のとおりである．

- 入力 1：マイクロコントローラ仕様（$\mathcal{A}$, $\mathcal{B}$, $A$, $\mathcal{C}$, $\mathcal{M}$, $\mathcal{F}$, $S$）．
- 入力 2：割当て要求をする周辺モジュール機能の列 $R = (f_0, f_1, ..., f_{|R|-1})$．ただし各 $0 \le t < |R|$ に対し $f_t \in \mathcal{F}$ である．
- 出力：割当て解すべての集合 $Z = \{z_0, z_1, ...\}$．ここで各 $z_i = (y_i^0, y_i^1, ..., y_i^t, ..., y_i^{|R|-1})$ は 1 つの割当て解で，以下の 5 つの制約すべてを満たす．なお $y_i^t$ $(0 \le t < |R|)$ は $f_t$ に対する周辺モジュールインスタンスとデバイスピンの割当てを表し，$y_i^t = (m_i^t, \{(a_{i,0}^t, b_{i,0}^t), (a_{i,1}^t, b_{i,1}^t), ...\}), m_i^t \in \mathcal{M}, a_{i,j}^t \in \mathcal{A}, b_{i,j}^t \in \mathcal{B}$ の形をとる．
- 制約 1：$\forall i \forall t : m_i^t \in M(f_t)$ — $m_i^t$ は，周辺モジュール機能 $f_t$ を提供する周辺モジュールインスタンスである．
- 制約 2：$\forall i \forall t : \{b_{i,0}^t, b_{i,1}^t, ...\} = B(f_t, m_i^t)$ — 各 $y_i^t$ には，該当周辺モジュールインスタンスの周辺ピンすべてが含まれており，しかも関係ない周辺ピンは含んでいない．
- 制約 3：$\forall i \forall t \forall j : a_{i,j}^t \in A(b_{i,j}^t)$ — 周辺ピン $b_{i,j}^t$ とデバイスピン $a_{i,j}^t$ は結合可能である．
- 制約 4：$\forall i \forall t \forall t' [t \ne t'] : m_i^t \ne m_i^{t'}$ — 各 $z_i$ の中では，1 つの周辺モジュールインスタンスが複数回割り当てられていない．
- 制約 5：$\forall i \forall t \forall t' \forall j \forall j' [(t \ne t') \vee (j \ne j')] : a_{i,j}^t \ne a_{i,j'}^{t'}$ — 各 $z_i$ の中では，どのデバイスピンも複数の周辺ピンに結合されない（重複結合がない）．□

**例 16** 入力の例（周辺モジュール機能の列）：$R = (f_0, f_1, f_2, f_3) = (\texttt{i2c}, \texttt{usart}, \texttt{usart}, \texttt{an})$ □

**例 17** 出力の例（割当て解の 1 つ）：$z_0 = (y_0^0, y_0^1, y_0^2, y_0^3)$, $y_0^0 = (\mathsf{I2C1}, \{(17, \mathsf{SCL1}), (18, \mathsf{SDA1})\})$, $y_0^1 = (\mathsf{U1}, \{(11, \mathsf{U1TX}), (12, \mathsf{U1RX})\})$, $y_0^1 = (\mathsf{U2}, \{(10, \mathsf{U2TX}), (14, \mathsf{U2RX})\})$, $y_0^2 = (\mathsf{AN}, \{(3, \mathsf{AN1})\})$ □

## 3. 個別デバイスピン割当て方式

本節では，デバイスピン個別に周辺ピンを結合する方式の（**個別デバイスピン割当て方式**）に基づくアルゴリズムを示す．定義 8 にあるとおり，周辺モジュール機能 $f$ に対して $S(f)$ は，$f$ を実現する周辺モジュールインスタンスと周辺ピンの割当てすべての集合を与える．割当て要求をする周辺モジュール機能の列 $R = (f_0, f_1, ..., f_{|R|-1})$ に対して，$t = 0, 1, ...$ の順で $f_t$ への割当てを $S(f_t)$ の要素から選び，割当て可能なら $f_{t+1}$ への割当てに進む．

### 3.1 バックトラック法

ここで示すアルゴリズム IdpBacktrack は，$R$ の先頭から順に可能な選択を再帰的に試みて，割当て不可能ならバックトラックして別の選択に移る（**図 2** 参照）．解が見つかる度に出力することで解の列挙を行う．自明なアルゴリズムであるが，後の議論のため概要を以下に示す．

以下の変数を使用する．

- $t$ $(= 0, 1, ..., |R| - 1)$：探索の再帰の深さを表す．
- $xm$ $(\subseteq \mathcal{M})$：割当て済みの周辺モジュールインスタンスの集合．
- $xdp$ $(= \{..., (a_i, b_i), ...\} \subseteq \mathcal{A} \times \mathcal{B})$：割当て済みの周辺モジュールピンとデバイスピンの結合の集合．

アルゴリズムは以下のとおりである[*3]．

```
procedure IdpBacktrack() {
  IDPBacktrackRec(0, ∅, ∅);
}
procedure IdpBacktrackRec(t, xm, xdp) {
  if (t = |R|) {   // 割当て成功
```

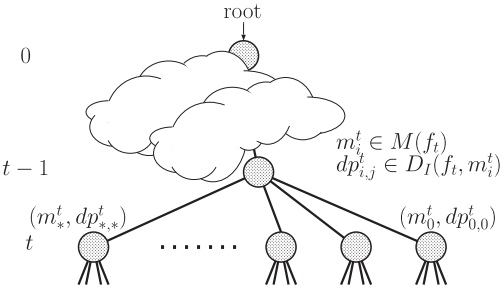



**図 2** バックトラック法
**Fig. 2** Backtrack method.

[*3] アルゴリズムの正しさは自明なので，証明は省略する．

```
    print xm, xdp;  // 解を表示
  } else {
    f_t := R[t];
    for each m ∈ M(f_t) ただし
        ∀m' ∈ xm : m' < m {
      for each dp ∈ D_I(f_t, m) ただし
          IdpExtendOk(xdp, dp) が真 {
        IdpBacktrackRec(t + 1, xm ∪ {m}, xdp ∪ dp);
      }
    }
  }
}
function IdpExtendOk(xdp, dp) {
// xdp と dp にデバイスピンの重複がなければ真
  return ({a_i : (a_i, b_i) ∈ xdp} ∩ {a_i : (a_i, b_i) ∈ dp} = ∅);
}
```

### 3.2 ZDD 構築法

解の列挙を表現する ZDD を構築するアルゴリズム IdpZdd の概略を述べる．各 $t = 0, 1, ..., |R| - 1$ に対し，順次 $f_t$ を割り当てていく点は IdpBacktrack と同様である．IdpZdd では，文献 [5] の simpath アルゴリズムと同様に，トップダウンかつ幅優先的に ZDD を構築していく．simpath はグラフを対象としているため，ZDD 構築はグラフの各辺に対して一様に加える/加えないの 2 者選択を行っている．一方，本論文での探索は，探索の各深さ $t = 0, 1, 2, ...$ に対して周辺モジュール機能 $f_t$ に関する選択を行う．この際，探索状態が等価な ZDD ノードがもし存在すれば，それを共有して探索の手数を削減する．IdpZdd は simpath と同様な構造なので掲載は省略する（付録 A.2 に示す）．

## 4. 互換デバイスピングループ割当て方式

本章では，結合に関して互換性のあるデバイスピン群をグループ化した**互換デバイスピングループ**（Compatible Device Pin Group; CDPG）の概念を提案する．この方法はデバイスピンを同値類分割し，同値類に対する周辺ピンの割当て探索を行う．

例を用いて基本的な考えを説明する．たとえば，6 本の周辺ピン $b_1, ..., b_6$ いずれもが再マップ可能で，4 本のデバイスピン $a_1, ..., a_4$ のいずれに対しても結合可能な場合を考える．個別デバイスピン割当て方式では，可能な組合せ数は ${}_6\mathrm{P}_4 = 360$ である．デバイスピン $a_1, ..., a_4$ は，周辺ピン $b_1, ..., b_6$ の結合可能性が互いに同じという意味において同等である．つまり $a_1, ..., a_4$ を 1 つのグループとし，このグループに 4 本までの周辺ピンが結合できると考えることができる．これが互換デバイスピングループ割当て方式の考え方である．今の例での可能な組合せ数は ${}_6\mathrm{C}_4 = 15$ であり，大きく減少している．

### 4.1 互換デバイスピングループ（CDPG）

互換デバイスピングループの定義を以下に示す．直感的にいえば，各周辺ピンに対する結合の可否が同一であるデバイスピンをグループ化したものである．条件 (2) はデバイスピンに対する同値関係を定め，条件 (3) はグループ化がその同値関係に基づく分割であることを表す．

**定義 18** 以下の条件をすべて満たすときおよびそのときのみに限り，$(\mathcal{G}, \{A_g : g \in \mathcal{G}\})$ を $\mathcal{A}'(\subseteq \mathcal{A})$ の $\mathcal{B}'(\subseteq \mathcal{B})$ に関する **CDPG 分割**と呼ぶ．

(1) $\forall g \in \mathcal{G} : \mathcal{G}_g \neq \emptyset$

(2) $\forall g \in \mathcal{G}\ \forall a, a' \in \mathcal{A}_g\ \forall b \in \mathcal{B}' : a \in A(b) \Leftrightarrow a' \in A(b)$

(3) $\{\mathcal{A}_g : g \in \mathcal{G}\}$ は $\mathcal{A}'$ の直和分割

各 $\mathcal{A}_g$ を**互換デバイスピングループ**あるいは **CDPG**，各 $g \in \mathcal{G}$ を**グループ番号**，$\mathcal{G}$ を**グループ番号の集合**と，それぞれ呼ぶ． □

自明な CDPG 分割の例を 2 つ示す．

**例 18** $(\emptyset, \emptyset)$ は $\emptyset$ の $\mathcal{B}$ に関する CDPG 分割である． □

**例 19** $\mathcal{G} = \mathcal{A}$ とし，各 $a \in \mathcal{A}$ に対して $\mathcal{A}_a = \{a\}$ とする．$(\mathcal{G}, \{A_g : g \in \mathcal{G}\})$ は $\mathcal{A}$ の $\mathcal{B}$ に関する CDPG 分割である． □

しかしこれらでは高速化は期待できそうもない．高速化の観点で重要な CDPG 分割は，グループ内のメンバ数が多いものと考えられる．この点に関して以下に 2 つの命題を示す．

最初の命題は，CDPG 分割で考慮する周辺ピンの集合を再マップ可能な周辺ピンの集合 $\mathcal{B}^R$（あるいはその部分集合）とすればよいことを示す．なぜなら，もし再マップ不可能な周辺ピン $b$ を考慮すると，$b$ と結合する唯一のデバイスピン $a$ の属するグループのメンバ数は必ず 1 となり，探索空間は小さくならないからである．

**命題 1** 再マップ不可能な任意の周辺ピン $b(\in \mathcal{B}^N)$，任意の $\mathcal{A}' \subseteq \mathcal{A}$，任意の $\mathcal{B}' \subseteq \mathcal{B}$（ただし $b \in \mathcal{B}'$）を考える．$\mathcal{A}'$ の $\mathcal{B}'$ に関する任意の CDPG 分割 $(\mathcal{G}, \{\mathcal{A}_g : g \in \mathcal{G}\})$ に対して，もしある $g \in \mathcal{G}$ と $a \in \mathcal{A}_g$ が存在して $\{a\} = A(b)$ ならば，$\{a\} = \mathcal{A}_g$ である．

**証明** 背理法で示す．ある $a'(\neq a)$ が存在して $\{a, a'\} \subseteq \mathcal{A}_g$ と仮定する．すると $a \in A(b)$ かつ $a' \in A(b)$ であり，$b$ が再マップ不可能という仮定 $b \in \mathcal{B}^N$ に矛盾する． □

次の命題で使用する記号 $\mathcal{A}^R$ と $\mathcal{A}^N$ を定義する．

**定義 19** $\mathcal{A}^R(\subseteq \mathcal{A})$ を再マップ可能な周辺ピンと結合可能な関係にあるデバイスピンすべての集合，すなわち $\mathcal{A}^R \stackrel{\text{def}}{=} \{a \in \mathcal{A} : b \in \mathcal{B}^R \wedge a \in A(b)\}$ と定める．$\mathcal{A}^N(\subseteq \mathcal{A})$ をそれ以外のデバイスピンすべての集合，すなわち $\mathcal{A}^N \stackrel{\text{def}}{=} \mathcal{A} \setminus \mathcal{A}^R$ と定める． □

次の命題は，CDPG 分割の対象とするデバイスピンの集合を $\mathcal{A}^R$（あるいはその部分集合）とすればよいことを示す．なぜなら，$\mathcal{A}^R$ に属さない（すなわち $\mathcal{A}^N$ に属する）デバイスピン $a$ を CDPG 分割の対象にすると $a$ の属する

グループのメンバ数は必ず 1 となり，探索空間は小さくならないからである．

**命題 2** 任意の $a \in \mathcal{A}^N$, 任意の $\mathcal{A}' \subseteq \mathcal{A}$(ただし $a \in \mathcal{A}'$)，任意の $\mathcal{B}' \subseteq \mathcal{B}$ を考える．$\mathcal{A}'$ の $\mathcal{B}'$ に関する任意の CDPG 分割 $(\mathcal{G}, \{\mathcal{A}_g : g \in \mathcal{G}\})$ に対して，もしある $g \in \mathcal{G}$ が存在して $a \in \mathcal{A}_g$ ならば，$\{a\} = \mathcal{A}_g$ またはすべての $b \in \mathcal{B}'$ に対し $a \notin A(b)$ である．

**証明** 背理法で示す．ある $a'(\neq a)$ が存在して $\{a, a'\} \subseteq \mathcal{A}_g$ かつ，ある $b \in \mathcal{B}'$ に対し $a \in A(b)$ と仮定する．すると $a \in A(b)$ かつ $a' \in A(b)$ であり，$b$ は再マップ可能な周辺ピンであることを意味する．すなわち $a$ に再マップ可能な周辺ピンが結合可能であり，仮定 $a \in \mathcal{A}^N$ に矛盾する．□

$\mathcal{A}^R$ の $\mathcal{B}^R$ に関する CDPG 分割を計算するアルゴリズム CdpgPart を以下に示す．このアルゴリズムの基本的な考え方は，結合可能な周辺ピンの集合（再マップ可能なもの）が互いに等しいデバイスピンどうしが 1 つの CDPG を構成する，というものである．

```
function CdpgPart(A^R, B^R) {
  // X_a : a と結合可能で再マップ可能な周辺ピンの集合
  for each a ∈ A^R : X_a := {b ∈ B^R : a ∈ A(b)};
  // 分割
  G := ∅; // グループ番号の集合
  g := 0; // 次のグループ番号
  for each a ∈ A^R : G_a := nil; // a のグループ番号
  for each a ∈ A^R {
    if (G_a = nil) { // a をグループの代表メンバにする
      // a および a と同等なものでグループを構成
      A_g := {a' ∈ A^R : X_a' = X_a};
      // グループ番号 g を割り当て
      for each a' ∈ A_g : G_a' := g;
      G := G ∪ {g};
      g := g + 1;
    }
  }
  return (G, {A_g, g ∈ G});
}
```

**命題 3** (i) CdpgPart は $\mathcal{A}^R$ の $\mathcal{B}^R$ に関する CDPG 分割を計算し，$\mathcal{G} \subseteq \mathcal{A}^R$ である．(ii) その動作時間は，たかだか $|\mathcal{A}|$ と $|\mathcal{B}|$ に関する多項式である．(iii) CdpgPart が求める CDPG 分割が真の細分となるような，$\mathcal{A}^R$ の $\mathcal{B}^R$ に関する CDPG 分割は存在しない．

**証明** (i) CDPG 分割の条件すべてが以下のとおり成立する．

- 条件 (1) 新たにグループ $\mathcal{A}_g$ を作るときには少なくとも 1 つの $a \in \mathcal{A}^R$ が含まれるので，$\mathcal{A}_g \neq \emptyset$ である．
- 条件 (2) $g \in \mathcal{G}$ と $a, a' \in \mathcal{A}_g$ を任意のものとし，以下では固定する．アルゴリズムにおける $X_a$ と $X_{a'}$ の構成方法より，各 $b \in \mathcal{B}^R$ に対して $b \in X_a \Leftrightarrow a \in A(b)$ であり，かつ $b \in X_{a'} \Leftrightarrow a' \in A(b)$ である．$a, a' \in \mathcal{A}_g$ より $X_a = X_{a'}$ なので，$\forall b \in \mathcal{B}^R : a \in A(b) \Leftrightarrow a' \in A(b)$ が成立する．
- 条件 (3) まず，任意の $g, g' \in \mathcal{G}$ $(g \neq g')$ に対して $\mathcal{A}_g \cap \mathcal{A}_{g'} = \emptyset$ が成立する．なぜなら，各 $a \in \mathcal{A}^R$ がある $\mathcal{A}_g$ に入ると，他の $\mathcal{A}_{g'}$ に入ることはないからである．次に，$\cup_{g \in \mathcal{G}} \mathcal{A}_g = \mathcal{A}^R$ が成立する．なぜなら，各 $g \in \mathcal{G}$ に対して $\mathcal{A}_g \subseteq \mathcal{A}^R$ であり，かつ各 $a \in \mathcal{A}^R$ はある $\mathcal{A}_g$ に属すからである．

(ii) 実行時間はアルゴリズム記述より明らかである．

(iii) ある CDPG 分割が存在して，CdpgPart の出力がその真の細分であると仮定する．すると，ある $a, a' \in \mathcal{A}^R$ $(a \neq a')$ が存在し，$a \in \mathcal{A}_g$ かつ $a' \in \mathcal{A}_{g'}$（ただし $g \neq g'$）であり，しかも $a$ と $a'$ は定義 18 の条件 (2) を満たす．一般性を失うことなく $g < g'$ と仮定する．するとアルゴリズム記述より $X_a = X_{a'}$ なので，$a$ が $\mathcal{A}_g$ に入るときに $a'$ も $\mathcal{A}_g$ に入るので，$g = g'$ である．これは仮定 $g \neq g'$ に矛盾する．□

### 4.2 派生的な記号の定義

探索アルゴリズムの記述で使用する記号 $D_G$ とそれに関連するものを定義する．前節の探索アルゴリズムで用いた $D_I$（定義 15 参照）の CDPG 版が $D_G$ であり，後で提案する探索アルゴリズムは $D_I$ の代わりに $D_G$ を用いる．

**定義 20** デバイスピン $a \in \mathcal{A}$ に対し，ある $g \in \mathcal{G}$ が存在して $a \in \mathcal{A}_g$ であるときおよびそのときのみに限り，$a$ を **CDPG デバイスピン**と呼ぶ．それ以外のとき，$a$ を**非 CDPG デバイスピン**と呼ぶ．□

周辺ピンの集合を 3 分割する．定義 10 では周辺ピンの集合を $\mathcal{B}^R$ と $\mathcal{B}^N$ の 2 つに分割した．さらに $\mathcal{B}^N$ を，結合するデバイスピンが CDPG デバイスピンであるか否かにより，$\mathcal{B}^{NC}$ と $\mathcal{B}^{NN}$ の 2 つに分割する．結果として $\mathcal{B}^R$, $\mathcal{B}^{NC}$, $\mathcal{B}^{NN}$ は $\mathcal{B}$ の直和分割となる．

**定義 21 CDPG デバイスピン結合の再マップ不可能な周辺ピンの集合**を $\mathcal{B}^{NC} \stackrel{\text{def}}{=} \{b \in \mathcal{B}^N : A(b) \subseteq \cup_{g \in \mathcal{G}} A_g\}$，**非 CDPG デバイスピン結合の再マップ不可能な周辺ピンの集合**を $\mathcal{B}^{NN} \stackrel{\text{def}}{=} \mathcal{B}^N \setminus \mathcal{B}^{NC}$ とそれぞれ定める．□

以下では，互換デバイスピン割当て方式のために，CDPG と周辺ピンの結合の集合を表す記号 $D_G(f, m)$ を定義する．個別デバイスピン割当て方式における定義 15 で示した $D_I(f, m)$ の各要素は $q = \{..., (a_i, b_i), ...\}$ の形であり，デバイスピンと周辺ピンとの結合を表していた．一方 $D_G(f, m)$ の各要素は $(\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$ の形であり，CDPG と周辺ピンとの結合を表す．3 つ組 $(g, B_g^R, B_g^{NC})$ は，CDPG $g \in \mathcal{G}$ に対して，再マップ可能な周辺ピン集合 $B_g^R$ と CDPG デバイスピン結合の再マッ

プ不可能な周辺ピン集合 $B_g^{NC}$ を結合することを表現する．$B^{NN}$ は，非 CDPG デバイスピンに結合する再マップ不可能な周辺ピンの集合を表す．

**定義 22** 周辺モジュール機能 $f \in \mathcal{F}$，周辺モジュールインスタンス $m \in \mathcal{M}$ に対し，CDPG と周辺ピンの結合すべての集合を以下のとおり定義する．

$$D_G(f,m) \stackrel{\text{def}}{=} \cup_{q \in D_I(f,m)} \{(dp(q), B^{NN}(q))\}$$
$$dp(q) \stackrel{\text{def}}{=} \{(g, B_g^R(q), B_g^{NC}(q)) : g \in \mathcal{G}\}$$

ただし $B_g^R(q)$，$B_g^{NC}(q)$，$B^{NN}(q)$ はそれぞれ以下のとおり定める．

- $B_g^R(q) \stackrel{\text{def}}{=} \{b \in \mathcal{B}^R : (a,b) \in q \wedge a \in \mathcal{A}_g\}$：$q$ の中に出現する再マップ可能な周辺ピンのうち，$\mathcal{A}_g$ 中のデバイスピンと結合するものの集合である．
- $B_g^{NC}(q) \stackrel{\text{def}}{=} \{b \in \mathcal{B}^{NC} : (a,b) \in q \wedge a \in \mathcal{A}_g\}$：$q$ の中に出現する CDPG デバイスピン結合の再マップ不可能な周辺ピンのうち，$\mathcal{A}_g$ 中のデバイスピンと結合するものの集合である．
- $B^{NN}(q) \stackrel{\text{def}}{=} \{b \in \mathcal{B}^{NN} : (a,b) \in q\}$：$q$ の中に出現する CDPG デバイスピン非結合の再マップ不可能な周辺ピンの集合である． □

上記の定義より $\{g_1, g_2, ...\} = \mathcal{G}$ とすれば，各 $q \in D_I(f,m)$ に対して $B_{g_1}^R(q)$，$B_{g_2}^R(q)$, ..., $B_{g_1}^{NC}(q)$, $B_{g_2}^{NC}(q)$, ..., $B^{NN}(q)$ は集合 $\{b : (a,b) \in q\}$ の直和分割となる．

**定義 23** $dp = (\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$ に対し，$dp.G \stackrel{\text{def}}{=} \{..., (g, B_g^R, B_g^{NC}), ...\}$, $dp.N \stackrel{\text{def}}{=} B^{NN}$ と，それぞれ定める． □

次に定義する演算 $\cup$ は，探索アルゴリズムでの割当ての拡大を行う際に用いる．

**定義 24** $dp = (\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$, $dp' = (\{..., (g, B_g'^R, B_g'^{NC}), ...\}, B'^{NN})$ に対し，二項演算 $\cup$ を $dp \cup dp' \stackrel{\text{def}}{=} (\{(g, B_g^R \cup B_g'^R, B_g^{NC} \cup B_g'^{NC}) : g \in \mathcal{G}, (g, B_g^R, B_g^{NC}) \in dp.G, (g, B_g'^R, B_g'^{NC}) \in dp'.G\}, dp.N \cup dp'.N)$ と定める． □

### 4.3 バックトラック法

CDPG の概念を用いたバックトラック法に基づく探索アルゴリズム CdpgBacktrack は，IdpBacktrack と同様に構成できる（誌面の都合により掲載は省略する）．違いは以下の 3 点である．

(1) デバイスピン個別への割当ての代わりに CDPG への割当てを行う．このとき，$D_I$ の代わりに $D_G$ を用い，拡大の際の演算 $\cup$ は定義 24 のものを用いる．

(2) 変数 $xdp$ および $dp$ はデータ型が異なり，$(\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$ の形で CDPG と周辺ピンの結合の集合を表す．なお各要素が表す値は定義 22 で説明したとおりである．

(3) 関数 IdpExtendOk の代わりに，以下に示す関数 CdpgExtendOk を用いる．この関数の中で，条件 $h_1$ は CDPG に割当て可能な周辺ピン数の上限を超えないことを，条件 $h_2$ は CDPG デバイスピンへの割当てがピン単位で重複しないことを，条件 $h_3$ は非 CDPG デバイスピンへの割当てがピン単位で重複しないことを，それぞれ表す．

```
function CdpgExtendOk(dp, dp') {
// dp に dp' を加えても割当てが重複しないとき
// およびそのときに限り，真を返す．ただし
//   dp = ({..., (g, B_g^R, B_g^NC), ...}, B^NN),
//   dp' = ({..., (g, B'_g^R, B'_g^NC), ...}, B'^NN) とおく
  bool h1 := ∀g ∈ G :
        |B_g^R| + |B_g^NC| + |B'_g^R| + |B'_g^NC| ≤ |A_g|;
  bool h2 := ∀g ∈ G ∀b, b' ∈ B_g^NC ∪ B'_g^NC :
        (b ≠ b') ⇒ (A(b) ≠ A(b'));
  bool h3 := ∀b, b' ∈ B^NN ∪ B'^NN :
        (b ≠ b') ⇒ (A(b) ≠ A(b'));
  return h1 ∧ h2 ∧ h3;
}
```

CDPG への割当てが行われたのち，具体的に周辺ピンとデバイスピンの結合を求める手順は以下のとおりである．

(1) $\mathcal{B}^{NN} \cup (\cup_{g \in \mathcal{G}} \mathcal{B}_g^{NC})$ に属する周辺ピン $b$：再マップ不可能なので，$A(b)$ にはデバイスピンが 1 つだけ含まれている．その唯一のデバイスピン $a \in A(b)$ が $b$ と結合する．

(2) 各 $g \in \mathcal{G}$ に対し $\mathcal{B}_g^R$ に属する周辺ピン $b$：再マップ可能であり，$\mathcal{A}_g$ の任意のデバイスピンと結合可能であるが，本手順のステップ (1) により再マップ不可能な周辺ピンが $\mathcal{A}_g$ と結合済みの場合がある．そのため，$\mathcal{A}_g$ 中の未結合のデバイスピンとの結合を行う．具体的には，$\mathcal{B}_g^R$ と $\mathcal{A}_g \setminus (\cup_{b \in \mathcal{B}_g^{NC}} A(b))$ との間でデバイスピンが重複しない任意の結合を選ぶ．なお，重複しない結合の組合せの列挙は各 CDPG $g \in \mathcal{G}$ の中だけで閉じており，容易に行える点に注意されたい．

### 4.4 ZDD 構築法

CDPG 方式で ZDD を構築するアルゴリズム CdpgZdd は，CdpgBacktrack で行った変更と同様な変更を IdpZdd に対して行うことで得られる．よって CdpgZdd の掲載は省略する（付録 A.3 に示す）．

## 5. 評価

本章では提案手法の有効性を，探索空間および探索時間の比較の 2 通りで行う．前者に関しては，組合せ数の概数見積りの計算で行う．後者に関しては，探索プログラムを実装して実験的評価を行う．個別ピン割当て方式の周辺モジュールインスタンスとデバイスピンの割当ての状況の管

表 2 実験対象のマイクロコントローラの概要（* 印：再マップ機能付き周辺ピンが付随）
Table 2 Outline of microcontrollers for evaluation (*: associated with remappable peripheral pins).

| デバイス名<br>パッケージ名 | デバイス<br>ピンの数 | 周辺モジュールインスタンス数（上段）<br>個別デバイスピン方式での選択肢数 $X_I$（中段）<br>CDPG 方式での選択肢数 $X_G$（下段） | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TMR* | IC* | OC* | USART* | SPI* | I²C | ADC | PORT | INT* | REFCLKO* | REFCLKI* |
| PIC32MX170F256B | 28 | 4 | 5 | 5 | 2 | 2 | 2 | 10 | 21 | 4 | 1 | 1 |
| | | 20 | 25 | 25 | 50 | 90 | 2 | 10 | 21 | 20 | 5 | 5 |
| SPDIP | | 4 | 5 | 5 | 2 | 4 | 2 | 10 | 21 | 4 | 1 | 1 |
| PIC32MX370F512H | 64 | 4 | 5 | 5 | 4 | 2 | 2 | 28 | 53 | 4 | 1 | 1 |
| | | 40 | 50 | 48 | 379 | 441 | 2 | 28 | 53 | 40 | 9 | 10 |
| QFN | | 8 | 10 | 8 | 18 | 16 | 2 | 28 | 53 | 8 | 1 | 1 |
| PIC32MZ2048ECM144 | 144 | 8 | 9 | 9 | 6 | 6 | 5 | 48 | 120 | 4 | 3 | 3 |
| | | 106 | 119 | 118 | 1,188 | 2,150 | 5 | 48 | 120 | 53 | 41 | 41 |
| TQFP | | 8 | 9 | 9 | 7 | 12 | 5 | 48 | 120 | 4 | 3 | 3 |

理には，ビットベクトルによる表現とビット演算を用いた実装が可能であり，高速に動作する．一方，CDPG 方式はオーバヘッドの大きい素朴な実装方法しか現時点では分かっていない．このため，実装したプログラムの実行時間を測定し，実証的に有効性を確認する．

### 5.1 実装と実験環境

本論文で提案するアルゴリズム 4 種類（バックラック法と ZDD 構築法それぞれに対する個別デバイスピン割当て方式と CDPG 方式）を C 言語で実装した．また，ZDD 演算パッケージは独自の実装を行った．なお CDPG 分割を行う際は，割当て要求に関係する周辺ピンとデバイスピンのみを対象とした．

アルゴリズムは以下の環境で実装し，実証実験を行った．

- 計算機：Lenovo S30
  - CPU：Intel Xeon E5-1620（クロック 3.60 GHz）
  - 主記憶：32 GByte
- OS：FreeBSD 9.2-Release amd64
- 言語処理系：GCC C Compiler 4.2.1

プログラム実行の際の使用リソースの上限は主記憶を 24 GByte，実行時間を 3,600 秒に設定し，これら上限を超過した場合は実行を強制終了した．

### 5.2 実験内容

評価は**表 2** に記載している規模の異なる 3 種のマイクロコントローラ [6], [7], [8] を対象とした．各周辺モジュールの詳細はデータシートを参照されたい．この表では各マイクロコントローラの各周辺モジュールクラスに対して，周辺モジュールインスタンスの数，対応する周辺モジュール機能での探索の際の選択肢数を個別デバイスピン方式の場合（後述の $X_I(f)$ の値）と CDPG 方式の場合（後述の $X_G(f)$ の値）それぞれで示している[*4]．たとえば PIC32MX170F256B に対する周辺モジュール機能 `oc` の場合，個別デバイスピン方式での周辺モジュールインスタンスと再マップの組合せは 25 通りあり[*5]，CDPG 方式での周辺モジュールインスタンスとデバイスピングループの組合せは 5 通りある．

評価の際のテストケースとして，以下に示す周辺モジュール機能の列 $R_\alpha$ と $R_\beta$ を用いた[*6]．

$$R_\alpha = (\texttt{ic}, \texttt{ic}, \texttt{ic}, \texttt{oc}, \texttt{oc}, \texttt{oc}, \texttt{oc}, \texttt{int}, \texttt{refclko}, \texttt{refclki}, \texttt{spi}, \texttt{spi}, \texttt{usart}, \texttt{usart})$$

$$R_\beta = (\texttt{ic}, \texttt{oc}, \texttt{oc}, \texttt{oc}, \texttt{oc}, \texttt{oc}, \texttt{usart}, \texttt{i2c}, \texttt{an}, \texttt{an}, \texttt{port}, \texttt{port}, \texttt{port}, \texttt{port}, \texttt{port}, \texttt{port}, \texttt{port}, \texttt{pge})$$

$R_\alpha$ では，いずれの周辺モジュール機能も再マップ可能な周辺ピンが付随するものを選び，CDPG の効果の観察を目的とする．一方 $R_\beta$ はある組み込みシステムの開発で実際に使用したもので，具体的な応用例における CDPG の

*4 TMR と INT に関して，tmr1 と int1 は特別な機能を有し，それぞれ tmr2,... と int2,... と完全に同一ではない．そのため tmr1 と int1 は別の周辺モジュールクラスとして除外し，表 2 の数値には含めていない．

*5 対応する周辺モジュールインスタンスは 5 つある．それぞれ 1 本の周辺ピンを持ち，5 つのデバイスピンに再マップされる．

*6 これら周辺モジュール機能の仕様の概略は以下のとおりである．なお $i = 1, 2, ...$ は周辺モジュールインスタンスの番号を表す．`ic` は周辺モジュール IC のインスタンス IC$i$ と周辺ピン IC$i$ を使用する．`oc` は周辺モジュール OC のインスタンス OC$i$ と周辺ピン OC$i$ を使用する．`int` は周辺モジュール INT のインスタンス INT$i$ と周辺ピン INT$i$ を使用する．`refclko` は周辺ピン REFCLKO または REFCLKO$i$ を使用する．`refclki` は周辺ピン REFCLKI または REFCLKI$i$ を使用する．`spi` は周辺モジュール SPI のインスタンス SPI$i$ と周辺ピン SCK$i$，SCL$i$，SDO$i$ を使用する．`usart` は周辺モジュール USART のインスタンス U$i$ と周辺ピン U$i$TX，U$i$RX を使用する．`pge` はマイクロコントローラへのプログラム書き込みとデバック機能を表し，複数あるチャンネル ($i$) の中から 1 つを選び，周辺ピン PGE$i$，PGD$i$ を使用する．

効果の観察を目的とする．なお $R_\beta$ に含まれる i2c，an，port，pge に付随する周辺ピンはいずれも再マップ不可能であり，CDPG の効果は出ない．

以下の説明において，記号 $R_\alpha^k$ により，長さ $k$ の $R_\alpha$ の接頭辞を表すものと定める．$R_\beta^k$ に対しても同様に定める．たとえば $R_\beta^3 = (\mathtt{ic}, \mathtt{oc}, \mathtt{oc})$ である．

### 5.3 実験結果 1：探索空間の大きさ

CDPG 導入による探索空間の大きさの減少を評価する．厳密な意味での探索空間の実際の大きさは探索アルゴリズムにより異なるため，探索アルゴリズムとは独立した形で概数見積りを行い比較する．以下に示すものは，対称性のある割当てや周辺モジュールインスタンスの重複割当てをも含んだ組合せ数を勘定したもので，実際の探索アルゴリズムはこれらすべてを探索するとは限らない点に注意されたい．

任意の周辺モジュール機能の列を $R = (f_0, f_1, ..., f_{|R|-1})$ とする．この列での探索空間の大きさを，各 $f_i$ への選択肢数の積により見積もる．個別ピン割当て方式における探索空間の大きさの見積り $X_I(R)$ は以下の式で与えられる．

$$X_I(R) = \sum_{1 \le k \le |R|} \prod_{0 \le t < k} X_I(f_t)$$

$$X_I(f) = \sum_{m \in M(f)} |D_I(f, m)|$$

CDPG 方式も同様であり，上の式の $D_I$ を $D_G$ で置き換えることで探索空間の大きさの見積り $X_G(R)$ が与えられる．

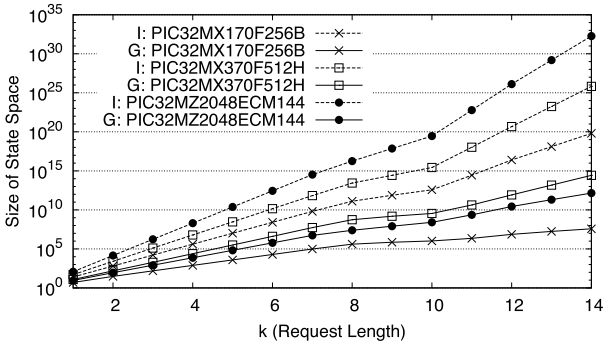
**図 3** に，テストケース $R_\alpha$ と $R_\beta$ に対する値を示す．横軸 $k$ に対し，図 3 (a) での縦軸は $X_I(R_\alpha^k)$ と $X_G(R_\alpha^k)$ の値であり，図 3 (b) での縦軸は $X_I(R_\beta^k)$ と $X_G(R_\beta^k)$ の値である．

- テストケース $R_\alpha$：図 3 (a) — PIC32MX170F256B の $k = 14$ の場合，探索空間の大きさの比は $1.7 \times 10^{12}$ である．規模がより大きな他のプロセッサでは，さらに大きい値となっている．
- テストケース $R_\beta$：図 3 (b) — PIC32MX170F256B の $k = 18$ の場合，比は $3.9 \times 10^5$ である．このテストケースは CDPG の効果が出にくいため，$R_\alpha$ の場合と比較すると比の値は小さい．

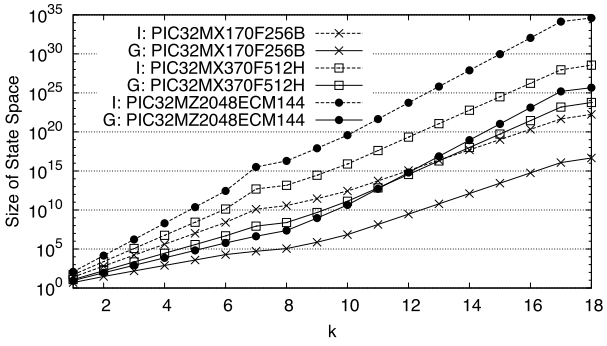
以上のことから，再マップ機能を有する周辺モジュールを多く用いる場合，CDPG の導入により探索空間の大きさが著しく減少することが示せた．

### 5.4 実験結果 2：探索時間

実装したプログラムの実行時間の測定結果を示す．各 $k = 1, 2, 3, ...$ に対して周辺モジュール機能の列 $R_\alpha^k$ および $R_\beta^k$ を割当て要求を行う．CDPG 方式では，CDPG への割当ての列挙を求めるのみとする．すなわち，得られた




(a) テストケース $R_\alpha$




(b) テストケース $R_\beta$

**図 3** CDPG 方式（G; 実線）と個別デバイスピン方式（I; 破線）の探索空間の大きさの比較

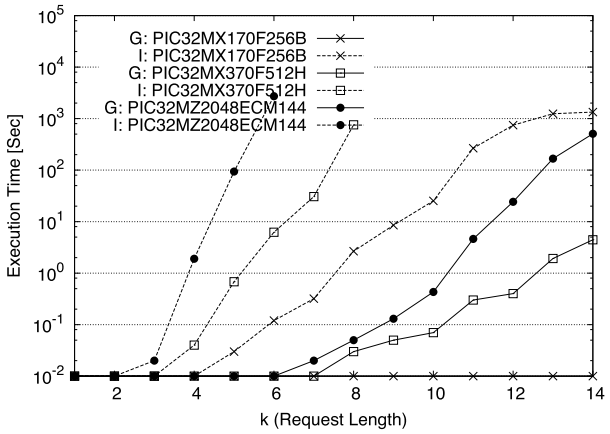
**Fig. 3** Comparison of size of search space: CDPG method (G; solid line) and Individual device pin method (I; dashed line).

CDPG への割当てから，さらに個別デバイスピンへの割当てを求めることは行わない．また CdpgPart による CDPG 分割は実行時に行い，測定結果はその実行時間も含めた値を示しているが，探索時間と比べて無視できる程度の短さである．

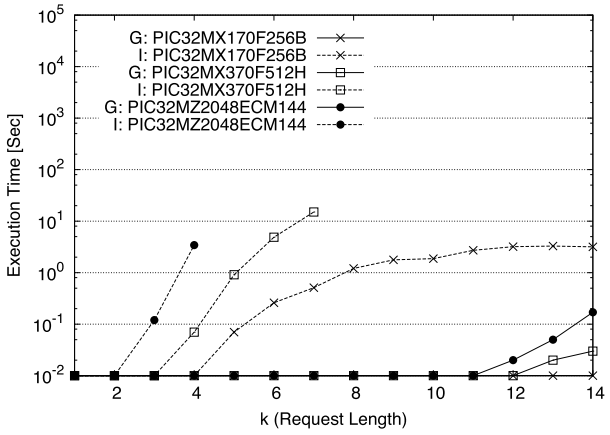
**図 4** にテストケース $R_\alpha$ での場合を，**図 5** にテストケース $R_\beta$ での場合を，それぞれ 5 回の平均値で示す[*7]．実行時間が 0.01 秒未満の場合，グラフでは 0.01 秒として示している．また，資源超過のため実行打ち切りとなった場合はグラフには示していない．なお資源超過の理由は，バックトラック法ではいずれも実行時間の超過で，ZDD 構築法ではいずれも使用メモリの超過である．

- テストケース $R_\alpha$：図 4 — PIC32MX170F256B の $k = 14$ の場合，バックトラック法および ZDD 構築法いずれも CDPG 方式は 0.01 秒以下であり，速度比はそれぞれ $10^5$ 以上，$10^2$ 以上である．PIC32MX370F512H の $k = 8$ の場合，バックトラック法での速度比は $2.5 \times 10^4$ である．他の場合においては，CDPG 方式が 0.01 秒以下のときに個別デバイスピン方式は資源

[*7] 提案アルゴリズムはいずれも決定性であるため，実行時間は実行の度に変化することはない．しかし測定には Unix での time コマンドを用いており，これに測定誤差が考えられることから，複数回の測定をする目的で複数回の実行を行った．



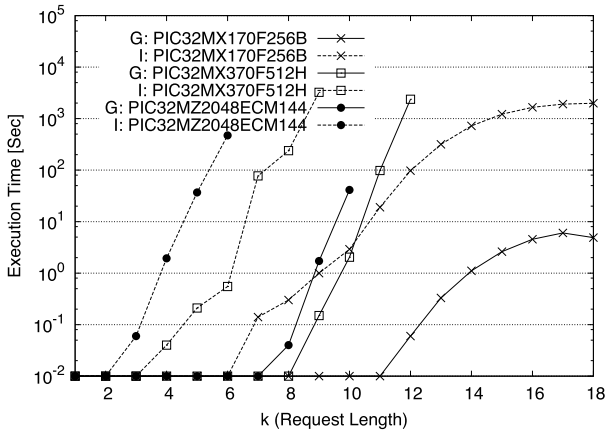

(a) バックトラック法



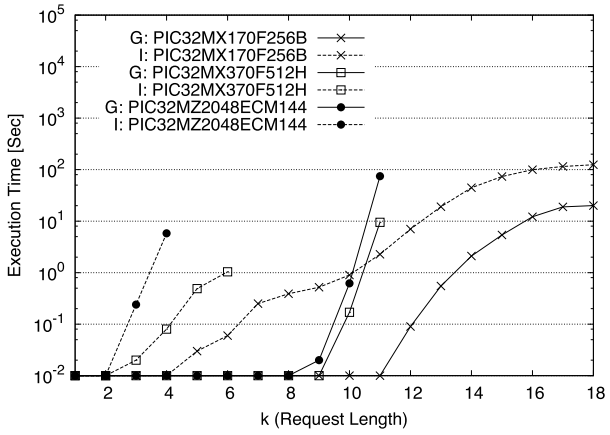

(b) ZDD 構築法

**図 4** CDPG 方式（G; 実線）と個別デバイスピン方式（I; 破線）の実行時間の比較（テストケース $R_\alpha$）

**Fig. 4** Comparison of running time (Test case $R_\alpha$): CDPG method (G; solid line) and Individual device pin method (I; dashed line).




(a) バックトラック法




(b) ZDD 構築法

**図 5** CDPG 方式（G; 実線）と個別デバイスピン方式（I; 破線）の実行時間の比較（テストケース $R_\beta$）

**Fig. 5** Comparison of running time (Test case $R_\beta$): CDPG method (G; solid line) and Individual device pin method (I; dashed line).

超過となり，著しい高速化を得ている．

- テストケース $R_\beta$：図 5 — このテストケースは CDPG 方式の効果が少ないために高速化の度合いは低いが，同様な傾向の結果が得られた．特に PIC32MX170F256B の $k = 18$ の場合で，速度比はバックトラック方式で $4.1 \times 10^2$，ZDD 構築法で 6.2 である．PIC32MX370F512H の $k = 9$ の場合，バックトラック法での速度比は $2.2 \times 10^4$ である．ほかの場合においては，CDPG 方式が 0.01 秒以下のときに個別デバイスピン方式は資源超過となり，著しい高速化を得ている．なお，再マップ不可能なデバイスピンが付随する周辺モジュールの探索（各 $k \geq 8$ の場合）が行われると，急速に速度比が低下している．

以上のことから，CDPG 方式はオーバヘッドが大きいが，再マップ機能を有する周辺モジュールを多く使用する場合に大きな高速化が達成でき，有用性を実証的に示せた．

## 6. おわりに

本論文では，組込システム用マイクロコントローラのための内蔵周辺モジュール割当てを高速化する CDPG 手法を提案し，有効性を実証的に示した．提案手法はデバイスピンを同値分割し，周辺ピンの割当てを同値類に対して行うことで組合せ数の減少を行う．この手法により大幅な高速化を得た．本論文で示した列挙アルゴリズムに加え，解を 1 つだけ発見して停止するアルゴリズムを，偽陽性を有する訪問済み状態記憶方式であるビット状態ハッシングおよびブルームフィルタ各方式を使用して実装した．これらにおいても同様な高速化を得たが，詳細は別稿に譲る．本論文の提案手法はアプリケーションドメイン固有の知識に基づいたものである．一方，問題を一般的な形で記述して，既存の各種手法を適用する方法も考えられる．たとえば，割当て制約を論理式として一般的な形で表現し，SAT ソルバ [3], [12] あるいは規約 BDD 計算法 [1] を使用して解を求

める方法もありうる．これら手法と本論文の提案手法との比較検討は今後の課題とする．

小規模なマイクロコントローラ以外の場合では資源超過のため実行打ち切りとなった．今後の課題は，大規模なマイクロコントローラでも列挙を可能にする方法の開発である．

### 参考文献


[1] Bryant, R.E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Computers*, Vol.C-35, No.8, pp.677–691 (1986).
[2] Clarke Jr., E.M., Grumberg, O. and Peled, D.: *Model Checking*, The MIT Press (1999).
[3] Eén, N. and Sörensson, N.: The MiniSat Page (online), available from ⟨http://www.minisat.se⟩ (accessed 2016-01-06).
[4] Holzmann, G.J.: *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley (2003).
[5] Knuth, D.E.: *The Art of Computer Programming 4A*, Addison-Wesley (2011).
[6] Microchip Technology Inc.: PIC32MX1XX/2XX Family Data Sheet DS60001168F (2014).
[7] Microchip Technology Inc.: PIC32MZ Embedded Connectivity (EC) Family Data Sheet, DS60001191D (2014).
[8] Microchip Technology Inc.: PIC32MX330/250/370/430/450/470 Data Sheet, DS60001185D (2015).
[9] Minato, S.: Zero-suppressed BDDs for set manipulation in combinatorial problems, *The 30th Conference on Design Automation*, pp.272–277 (1993).
[10] Morris, R.: Scatter strage techniques, *Comm. ACM*, Vol.11, No.1, pp.38–44 (1968).
[11] The OEIS Foundation Inc.: The On-Line Encyclopedia of Integer Sequences, A007764 Number of nonintersecting (or self-avoiding) rook paths joining opposite corners of an n X n grid (online), available from ⟨http://oeis.org⟩ (accessed 2015-05-07).
[12] Yinlei Yu, Pramod Subramanyan, N.T. and Malik, S.: All-SAT using Minimal Blocking Clauses, *Proc. VLSI Design.* (*VLSID*), Mumbai, India, pp.86–91 (2014).
[13] 川原 純，斎藤寿樹，鈴木 拡，湊 真一：ZDD によるパスの列挙，数理解析研究所講究録，Vol.1744, pp.35–41 (2011).

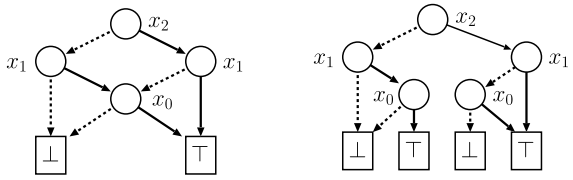

## 付 録

## A.1 ZDD の概要

**ZDD** は集合族をメモリ効率良く表現するデータ構造であり，BDD（Binary Decision Diagram; 二部決定グラフ）の亜種である [1], [5], [9]．ZDD では以下の 3 種類のノードが用いられる（以下ではこれら 3 種類を総称して **ZDD ノード**と呼ぶ）．

- 空集合を表す終端ノード ⊥
- 単位集合を表す終端ノード ⊤
- 内部ノード（以下では **ZDD 内部ノード**と呼ぶ）

各 ZDD 内部ノードは，変数名，1 枝，0 枝の情報を持つ．1 枝および 0 枝は他の ZDD ノードを指すポインタである．




(a) 既約 ZDD　　(b) 非既約 ZDD

図 **A·1** 集合族 $\{\{x_1, x_0\}, \{x_2, x_0\}, \{x_2, x_1\}\}$ を表現する ZDD の例

**Fig. A·1** Examples of ZDDs that represent a family of sets $\{\{x_1, x_0\}, \{x_2, x_0\}, \{x_2, x_1\}\}$.

図示する場合には，**図 A·1** に示すとおり，1 枝は実線の有向辺で，0 枝は点線の有向辺でそれぞれ描き，変数名は ZDD 内部ノードに添えて（あるいは中に）書く．

ZDD が表現する集合族は，以下の規則に従う．

- 最上位の ZDD ノードから ⊤ への経路が 1 つの集合を表す．
- 各経路において 1 枝を出発する ZDD 内部ノードの変数およびそれのみに限り，その経路に対応する集合に含まれる．

ZDD には，BDD とは異なる簡約化規則が定められている（簡約化規則の詳細は文献 [9] を参照されたい）．与えられた集合族を表現する ZDD は一般に複数存在し得るが，簡約化規則を繰り返し適用することにより，（ある与えられた変数の順序のもとでの）ZDD 内部ノード数最小かつ一意な表現，すなわち既約形が得られる．

図 A·1 に ZDD の例を示す．いずれも集合族 $\{\{x_1, x_0\}, \{x_2, x_0\}, \{x_2, x_1\}\}$ を表すものであり，図 A·1 (a) は既約，図 A·1 (b) は非既約である．

## A.2 個別デバイスピン方式 ZDD 構築法

個別デバイスピン方式で ZDD を構築するアルゴリズムを示す．本アルゴリズムでは，割当て要求 $R = (f_0, f_1, ...) \in \mathcal{F}^*$ の先頭から順に，$f_t$ $(t = 0, 1, ...)$ を実現する周辺モジュールインスタンスとデバイスピンの結合の割当てを列挙する．ZDD の構築は，文献 [5] での simpath アルゴリズムと同様に幅優先で行う．simpath ではグラフの各辺をパスに加える/加えないの選択を行っているのに対し，IdpZdd では，各深さ $t = 0, 1, ...$ で $f_t$ に対する周辺モジュールインスタンスとピン結合の選択を行っている．もし各選択で制約に違反する場合，ZDD に組み込まないことで ZDD サイズを抑制する．詳しくは以下のとおりである．

- 構築する ZDD 内の各 ZDD 内部ノードは，対応する $f_t$ $(0 \le t < |R|)$ を表す整数 $t$ で分類される．この整数値のことを**深さ**と呼ぶ．深さ $t$ の各 ZDD 内部ノードは，$f_t$ に対する周辺モジュールインスタンスとデバイスピンの結合の 1 つを表す．そして ZDD 内部ノードの 1 枝は，この結合を選択することを表す．したがっ

て，ZDD 内部ノード $z$ の深さ $t$ は，ZDD の根から $z$ へ到達するときに経由する 1 枝の数に一致する．
アルゴリズム記述の簡単化のため，**図 A·2** (a) に示すとおり，深さ $-1$ には構築作業用のダミーの ZDD 内部ノードを置くものとする．

- 根から ⊤ への 1 つの経路は $R$ を満たす 1 つの割当て解に対応する．経路中の ZDD 内部ノードのうちで 1 枝を出発するものに対応する結合が，割当て解を構成する周辺モジュールインスタンスとデバイスピンである．
- 深さ $t$ の ZDD 内部ノードは変数 $N[t]$ に蓄積し，等価な ZDD 内部ノードの検索の対象とする．そして等価な ZDD 内部ノードの部分 ZDD を共有し，探索空間の再探索を避けることで，探索コストの削減を図る．等価性と共有の詳細は後述する．

各 ZDD 内部ノードには以下の情報を持たせる．

- *var*：ZDD 内部ノードの変数名で，$\langle t, m, dp\rangle$ の形式とする[*8]．
  - $t$ $(= 0, 1, ...)$：この ZDD 内部ノードの深さ．
  - $m$ $(\in \mathcal{M})$：この ZDD 内部ノードで割当てをする周辺モジュールインスタンス．
  - $dp$ $(\subseteq \mathcal{A} \times \mathcal{B})$：この ZDD 内部ノードで割当てをするデバイスピンと周辺ピンの結合の集合．

  なお，ZDD 内部ノード $z$ に対し，$z.var.t$，$z.var.m$，$z.var.dp$, それぞれにより, 変数名 $z.var$ の中の各フィールドの値を表すものとする．
- *branch0*，*branch1*：この ZDD 内部ノードの 0 枝，および 1 枝．
- $xm$ $(\subseteq \mathcal{M})$：根からこの ZDD 内部ノードまでで割当てをした周辺モジュールインスタンスの集合（この値と根からこの ZDD 内部ノードの間の ZDD 内部ノードでの $var.m$ の値は，互いに整合が取れている）．
- $xdp$ $(\subseteq \mathcal{A} \times \mathcal{B})$：根からこの ZDD 内部ノードまでで割当てをしたデバイスピンと周辺ピンの集合（この値と根からこの ZDD 内部ノードの間の ZDD 内部ノードでの $var.dp$ の値は，互いに整合が取れている）．

ZDD 内部ノードの等価性の定義を述べる．基本的な考えは，探索途中にある 2 つの ZDD 内部ノード $z$ と $z'$ それぞれでのその後の探索において，新たなデバイスピンと周辺モジュールインスタンスの割当ての可否が互いに同値であれば，$z$ と $z'$ を同値とする，というものである．提案アルゴリズム IdpZdd では，以下に示す定義を用いる．

**定義 25** 2 つの ZDD 内部ノード $z$ と $z'$ が **I 等価**である

[*8] 本論文では，ZDD の変数名の順序に関する制約として，親ノードは子ノードよりも小さな順序を持つものとする．すなわち，根ノードの変数名は一番小さな順序である．具体的には，変数間の二項関係 $<$ は，$\langle t, m, dp\rangle < \langle t', m', dp'\rangle$ と $(t < t') \vee ((t = t') \wedge (m < m')) \vee ((t = t') \wedge (m = m') \wedge (dp < dp'))$ が同値と定める．なお，この関係は変数間の全順序を定める．

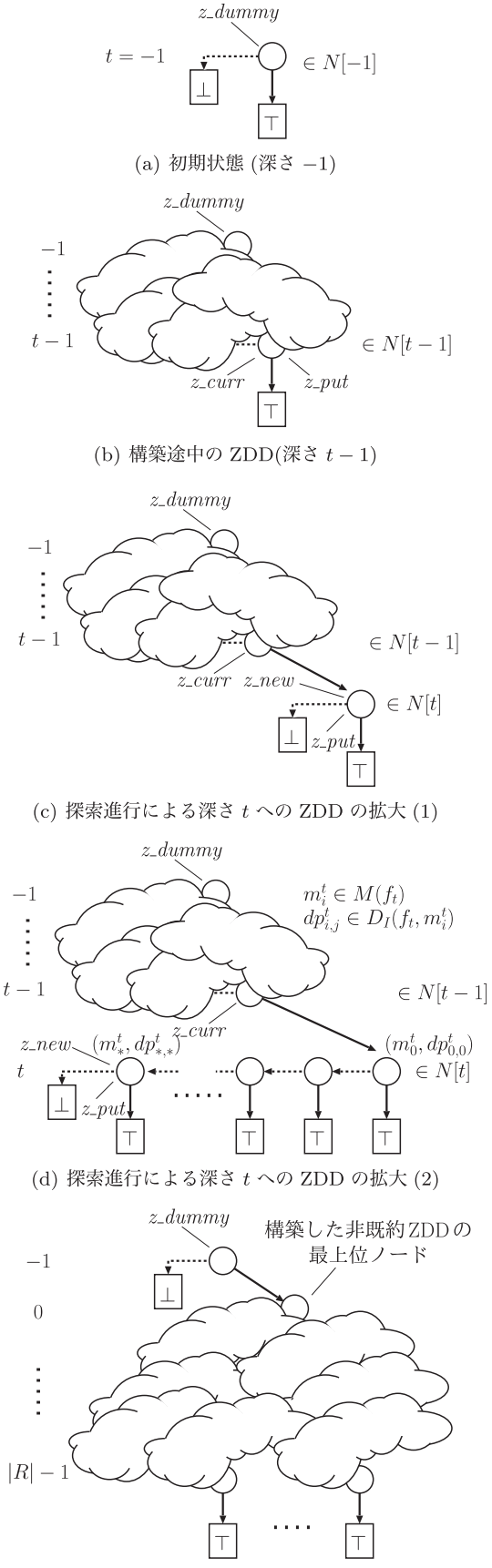



**図 A·2** ZDD 構築の方法
**Fig. A·2** The method to construct a ZDD.

とは，以下の条件すべてが成立するときおよびそのときのみに限る．なお $z$ で割り当てられている周辺モジュールインスタンスの集合を $\{m_0, m_1, ...\}$，デバイスピンと周辺ピンの結合を $\{..., (a, b), ...\}$ とし，$z'$ ではそれぞれ $\{m'_0, m'_1, ...\}$，および $\{..., (a', b'), ...\}$ とする．

(1) ともに同じ深さである（以下，深さを $t$ とおく）．

(2) 周辺モジュールインスタンスの割当てに関して，$M_t(z) = M_t(z')$ が成立する．ただし $M_t(z) = \{m \in \{m_0, m_1, ..., m_t\} : \text{ある } i > t \text{ に対し } C(m) = C(f_i)\}$ と定め，$M_t(z')$ も同様に定める（今後探索される周辺モジュールインスタンスと同じクラスに属する割当て済みの周辺モジュールインスタンスの集合が，$z$ と $z'$ で同一である）．

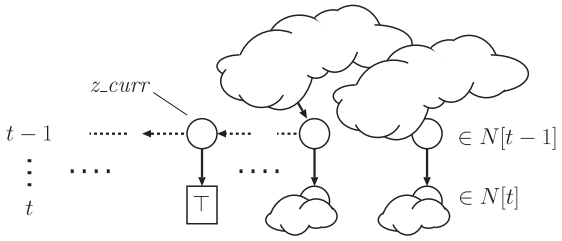
(3) デバイスピンの割当てに関して，$\{..., a, ...\} = \{..., a', ...\}$ が成立する（割当て済みのデバイスピン集合が，$z$ と $z'$ で同一である）．

ZDD 内部ノード $z$ と $z'$ が I 等価であるときおよびそのときのみに限り，$z \overset{\mathrm{I}}{\equiv} z'$ と表記する． □

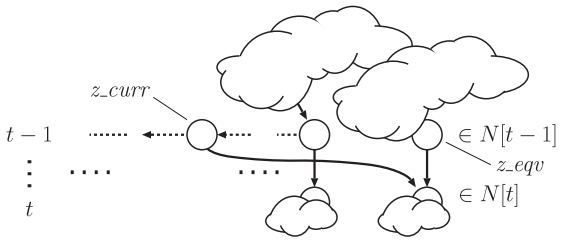
このI等価は同値関係であるが，誌面の都合によりその証明は省略する[*9,*10]．

提案アルゴリズム IdpZdd を以下に示す．なお関数 IdpExtendOk は，アルゴリズム IdpBacktrack で使用したものと同じものを用いる．探索の際にI等価な ZDD 内部ノードが存在する場合，**図 A·3** に示すように部分 ZDD を共有する[*11]．

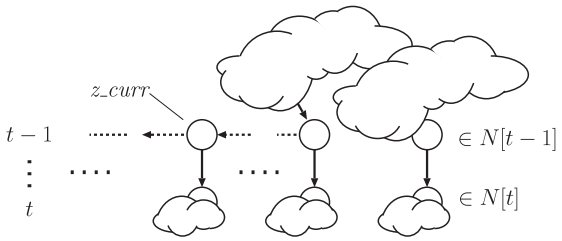
```
procedure IdpZdd() {
  // 非既約 ZDD を構築
  ZddNode z_dummy := ZddNode.new();
  ZddNodeSet N[−1 .. |R| − 1];
  N[t] := ∅,  t = −1, 0, ..., |R| − 1;
  // ZDD の初期構造：図 A·2 (a) 参照
  z_dummy.branch0 := ⊥;
  z_dummy.branch1 := ⊥;
  z_dummy.xm := ∅;
  z_dummy.xdp := ∅;
  N[−1].put(z_dummy);
  // ZDD を拡大
  for (t := 0; t < |R|; t := t + 1) {
    f_t := R[t];
```




(a) $z\_curr$ に対するI等価な ZDD 内部ノード検索前 (図 A·2 (b) 参照)




(b) I 等価な ZDD 内部ノード $z\_eqv$ が存在する場合




(c) 存在しない場合 (図 A·2(c), (d) 参照)

**図 A·3** I 等価な ZDD 内部ノードに対する部分 ZDD の共有
**Fig. A·3** Sharing a partial ZDD of I-equivalent ZDD nodes.

```
  // f_t に対する割当てを拡大
  for each z_curr ∈ N[t − 1] {
    // 図 A·2 (b), 図 A·3 (a) 参照
    if ∃z_eqv ∈ N[t − 1] :
        (z_eqv ≡^I z_curr) ∧ (z_eqv.branch1 ≠ ⊤) {
      // 部分 ZDD ノードを共有：図 A·3 (b) 参照
      z_curr.branch1 := z_eqv.branch1;
    } else { // 図 A·3 (c) 参照
      // z_put: 新規 ZDD 内部ノードの組み込み場所
      z_put := z_curr; // 図 A·2 (b) 参照
      z_curr.branch1 := ⊥;
      for each m ∈ M(f) ただし
          ∀m' ∈ z_curr.xm : m' < m {
        for each dp ∈ D_I(f_t, m) ただし
            IdpExtendOk(n_curr.xdp, dp) が真 {
          // 新規 ZDD 内部ノードを生成
          z_new := ZddNode.new();
          z_new.var := ⟨t, m, dp⟩;
          z_new.branch0 := ⊥;
          z_new.branch1 := ⊤;
          z_new.xm := z_curr.xm ∪ {m};
```

*9 同値関係であることの重要性は以下の理由による．$z \overset{\mathrm{I}}{\equiv} z'$ のときに，$z$ と $z'$ で ZDD 内部ノードを共有したとする．さらに $z' \overset{\mathrm{I}}{\equiv} z''$ であるときは，同値関係の推移律により $z \overset{\mathrm{I}}{\equiv} z''$ なので，$z, z', z''$ で ZDD 内部ノードを共有してよいからである．

*10 同値関係 $\overset{\mathrm{I}}{\equiv}$ は，ZDD が表現する集合族の等価性に基づく同値関係（$\overset{\mathrm{Z}}{\equiv}$ と表記）と必ずしも同一ではない．同値関係 $\overset{\mathrm{I}}{\equiv}$ は，同値関係 $\overset{\mathrm{Z}}{\equiv}$ の細分（または同一）である．同値関係 $\overset{\mathrm{I}}{\equiv}$ を提案アルゴリズム中で用いる理由は，探索途中の状態で同値判定を比較的高速に行えるからである．

*11 ハッシュ表を用いることで等価な ZDD 内部ノードの検索は高速に行える．

```
            z_new.xdp := z_curr.xdp ∪ dp;
            N[t].put(n_new);
            // 構築中の ZDD に組み込む
            if (z_put = z_curr) { // 図 A·2 (c) 参照
              z_put.branch1 := z_new;
            } else { // 図 A·2 (d) 参照
              z_put.branch0 := z_new;
            }
            z_put := z_new; // 組み込み場所を更新
          }
        }
      }
    }
  } // 図 A·2 (e) 参照
  // 既約 ZDD を求める
  return ZddReduce(z_dummy.branch1);
}
```

**命題 4** アルゴリズム IdpZdd が構築する ZDD は割当て要求 $R = (f_0, f_1, ...)$ を満たす解の列挙を表現する．

**証明** ($\Rightarrow$) 構築した ZDD が含む集合はいずれも，$R$ に対する割当て解に相当する．なぜなら，アルゴリズム構成上，(1) 各深さ $t$ に対して深さ $t$ の ZDD 内部ノードの 1 枝を 1 度だけ通り，(2) 深さ $t$ の ZDD 内部ノードは $f_t$ に対する周辺モジュールインスタンスとデバイスピンの結合の選択の 1 つであり，そして (3) 周辺モジュールインスタンスとデバイスピンは重複しないからである．

($\Leftarrow$) $R$ に対する割当て解に相当する集合は，構築した ZDD に必ず含まれる．もしある割当て解が含まれないと仮定すれば，ある深さ $t$ において解に対応する ZDD 内部ノードの 1 枝を通らない．しかし深さ $t$ での動作において，アルゴリズム構成上そのようなことは生じない． □

## A.3 CDPG 方式 ZDD 構築法

CDPG 方式で ZDD を構築するアルゴリズム CdpgZdd を提案する．提案アルゴリズムは IdpZdd に対して CDPG の概念を適用したものである．適用方法は CdpgBacktrack のときと同じなので，誌面の都合によりアルゴリズム全体の掲載は省略する．

各 ZDD ノードに対する CDPG と周辺ピンの結合の状況は，$(\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$ の形で表現される．2 つの ZDD ノードの等価性として，定義 25 で示した I 等価の CDPG 版である G 等価を以下に定義し，これを代わりに使用する．異なる点は条件 (3) のみである．

**定義 26** 2 つの ZDD ノード $z$ と $z'$ が **G 等価**であるとは，以下の 3 条件すべてが成立するときおよびそのときのみに限る．なお $z$ には周辺モジュールインスタンスの集合 $\{m_0, m_1, ...\}$，が，CDPG と周辺ピンの結合の状況 $(\{..., (g, B_g^R, B_g^{NC}), ...\}, B^{NN})$ が割り当てられ，$z'$ ではそれぞれ $\{m'_0, m'_1, ...\}$，および $(\{..., (g, B_g'^R, B_g'^{NC}), ...\}, B'^{NN})$ とする．

(1) ともに同じ深さである（以下，深さを $t$ とおく）．

(2) 周辺モジュールインスタンスの割当てに関して，$M_t(z) = M_t(z')$ が成立する．ここで $M_t(z)$ と $M_t(z')$ は，定義 25 でのものと同じである．

(3) CDPG への割当てに関して，以下の 3 条件が同時に成立する．ただし $A_g(B_g) = \{a \in \mathcal{A}_g : a \in A(b) \wedge b \in B_g\}$ とする．

- $\forall g \in \mathcal{G} : |B_g^R| = |B_g'^R|$
- $\forall g \in \mathcal{G} : A_g(B_g^{NC}) = A_g(B_g'^{NC})$
- $A_g(B^{NN}) = A_g(B'^{NN})$

ZDD ノード $z$ と $z'$ が G 等価であるときおよびそのときのみに限り，$z \overset{\mathrm{G}}{\equiv} z'$ と表記する． □

G 等価は同値関係であるが，誌面の都合によりその証明は省略する[*12]．

**角川 裕次** （正会員）

1990 年山口大学工学部電子工学科卒業．1992 年広島大学大学院工学研究科情報工学専攻博士課程前期修了．広島大学助手，講師，助教授を経て，現在，大阪大学大学院情報科学研究科准教授．分散アルゴリズムと教育工学の研究に従事．博士 (工学)．情報処理学会，IEEE Computer Society，電子情報通信学会各会員．

[*12] 同値関係 $\overset{\mathrm{G}}{\equiv}$ は ZDD ノードの同値関係 $\overset{\mathrm{Z}}{\equiv}$ の細分（あるいは同一）である．この定義を用いる理由は，同値判定が比較的高速に行えるからである．